```c
/*
 * SYTI_Test.c
 *
 * Created: 23.02.2023 10:44:53
 * Author : Manuel Strasser | 5CHIT
 */

#define F_CPU 16000000
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <util/atomic.h>
#include <util/delay.h>

//Zähler des Timers
volatile int timercnt=0;

//Button-Status -> 0: nicht gedrückt, 1: gedrückt
volatile int buttonStatus=0;

//Status, ob der Timer für das Warnsignal läuft -> 0: false, 1: true
volatile int iscountingForSignal = 0;

//Status, ob der Timer für die Notbremse läuft -> 0: false, 1: true
volatile int iscountingForStop = 0;

void init();
void init_timer();
void TurnOnLED();
void StopTrain();

int main(void)
{
	init();
	init_timer();

	//Interrupts aktivieren
	sei();

    while (1)
    {
		//Atomic Block, damit Interrupts während des Timer Counter auslesens,
unterdrückt werden
		ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
		{
			if (iscountingForSignal == 1 && timercnt >= 30) //überprüfen, ob der
Button 30s lang gedrückt wurde
			{
				iscountingForSignal = 0;
				iscountingForStop = 1;
				timercnt = 0;
				TurnOnLED();
			}

			if (iscountingForStop == 1 && timercnt >= 15) //überprüfen, ob
weitere 15s vergangen sind
			{
				iscountingForStop = 0;
				StopTrain();
```

```
                    }

                    if (buttonStatus == 1 && iscountingForSignal == 0) //wenn Button
gedrückt wird, das 30s Intervall starten
                    {
                           //Timer zurücksetzen, um ab jetzt 30 Sekunden zählen zu können
                           timercnt = 0;
                           iscountingForSignal = 1;
                    }
                    else if (buttonStatus == 0 && iscountingForSignal == 1) //wenn der
Button nicht mehr gedrückt wird, 30s Timer ignorieren
                    {
                           iscountingForSignal = 0;
                    }
             }

    }
}

//Register für LED, Button und Notbremse konfigurieren
void init()
{
       //LED Port -> PORTC2 auf Output -> HIGH: LED an, LOW: LED aus
       DDRC |= (1<<PORTC2);

       //Notbremse Port -> PORTD2 auf Output -> HIGH: Notbremse an, LOW: Notbremse aus
       DDRD |= (1<<PORTD2);

       //Button Interrupt Register setzen
       //PORT auf HIGH, für internen Pull Up
       PORTC |= (1<<PORTC1);
       //Pin Change Mask Register setzen
       PCMSK1 |= (1<<PCINT9);
       //Pin Change Interrupt aktivieren
       PCICR |= (1<<PCIE1);

}

//Register für 16bit Timer konfigurieren
void init_timer()
{
       // Timer CTC Modus aktivieren
       TCCR1B |= (1<<WGM12);
       // Prescaler auf 256
       TCCR1B |= (1<<CS12);
       // Output Compare A Interrupt aktivieren
       TIMSK1 |= (1<<OCIE1A);
       // Compare Wert setzen -> F_CPU *  1s / 256 (Prescaler) = 62500 Schritte um 1s
Intervall zu erreichen
       OCR1A = 62500;
}

//wird aufgerufen, wenn Warnsignal erfolgt
void TurnOnLED()
{
       //LED Port auf HIGH
       PORTC |= (1<<PORTC2);
}
```

```
//wird aufgerufen, wenn Notbremse erfolgt
void StopTrain()
{
	//Port für Notbremse auf HIGH
	PORTD |= (1<<PORTD2);
}

//Interrupt Routine für Button Interrupt
ISR(PCINT1_vect)
{
	//wenn Button gedrückt
	if(!(PINC & (1<<PINC1)))
	{
		buttonStatus = 1;
	}
	else
	{
		buttonStatus = 0;
	}
}

//Interrupt Routine für Timer Compare Interrupt
ISR(TIMER1_COMPA_vect)
{
	timercnt++; //Timer Counter wird somit alle 1s um 1 erhöht
}
```